SOLVING A LOW-RANK FACTORIZATION MODEL FOR MATRIX COMPLETION BY A NONLINEAR SUCCESSIVE OVER-RELAXATION ALGORITHM

ZAIWEN WEN [†‡] , WOTAO YIN[†] , AND YIN ZHANG[†]

Abstract. The matrix completion problem is to recover a low-rank matrix from a subset of its entries. The main solution strategy for this problem has been based on nuclear-norm minimization which requires computing singular value decompositions – a task that is increasingly costly as matrix sizes and ranks increase. To improve the capacity of solving large-scale problems, we propose a low-rank factorization model and construct a nonlinear successive over-relaxation (SOR) algorithm that only requires solving a linear least squares problem per iteration. Convergence of this nonlinear SOR algorithm is analyzed. Numerical results show that the algorithm can reliably solve a wide range of problems at a speed at least several times faster than many nuclear-norm minimization algorithms.

Key words. Matrix Completion, alternating minimization, nonlinear GS method, nonlinear SOR method

AMS subject classifications. 65K05, 90C06, 93C41, 68Q32

1. Introduction. The problem of minimizing the rank of a matrix arises in many applications, for example, control and systems theory, model reduction and minimum order control synthesis [20], recovering shape and motion from image streams [25, 30], data mining and pattern recognitions [6] and machine learning such as latent semantic indexing, collaborative prediction and low-dimensional embedding. In this paper, we consider the Matrix Completion (MC) problem of recovering a low-rank matrix from a subset of its entries, that is,

$$(1.1) \qquad \min_{W \in \mathbb{R}^{m \times n}} \; \mathrm{rank}(W), \; \text{s.t. } W_{ij} = M_{ij}, \; \forall (i,j) \in \Omega,$$

where $\mathrm{rank}(W)$ denotes the rank of W, and $M_{i,j} \in \mathbb{R}$ are given for $(i,j) \in \Omega \subset \{(i,j) : 1 \leq i \leq m, 1 \leq j \leq n\}$. Although problem (1.1) is generally NP-hard due to the combinational nature of the function $\mathrm{rank}(\cdot)$, it has been shown in [27, 3, 4] that, under some reasonable conditions, the solution of problem (1.1) can be found by solving a convex optimization problem:

$$(1.2) \qquad \min_{W \in \mathbb{R}^{m \times n}} \; \|W\|_*, \; \text{s.t. } W_{ij} = M_{ij}, \; \forall (i,j) \in \Omega,$$

where the *nuclear* or *trace* norm $\|W\|_*$ is the summation of the singular values of W. In particular, Candès and Recht in [3] proved that a given rank-r matrix M satisfying certain incoherence conditions can be recovered exactly by (1.2) with high probability from a subset Ω of uniformly sampled entries whose cardinality $|\Omega|$ is of the order $O(r(m + n)\mathrm{polylog}(m + n))$. For more refined theoretical results on matrix completion we refer the reader to [2, 4, 11, 14, 15, 26, 37].

Various types of algorithms have been proposed to recover the solution of (1.1) based on solving (1.2). One method is the singular value thresholding algorithm [13] using soft-thresholding operations on the singular values of a certain matrix at each iteration. Another approach is the fixed-point shrinkage algorithm [22] which solves the regularized linear least problem:

$$(1.3) \qquad \min_{W \in \mathbb{R}^{m \times n}} \; \mu\|W\|_* + \frac{1}{2}\|\mathcal{P}_\Omega(W - M)\|_F^2,$$

1

where $\mathcal{P}_\Omega$ is the projection onto the subspace of sparse matrices with nonzeros restricted to the index subset Ω. An accelerated proximal gradient algorithm is developed in [29] based on a fast iterative shrinkage-thresholding algorithm [1] for compressive sensing. The classical alternating direction augmented Lagrangian methods have been applied to solve (1.2) in [8, 33] and the closely related sparse and low-rank matrix decomposition in [34]. Other approaches include [16, 19, 24, 23, 5, 18]. All of those nuclear-norm minimization algorithms bear the computational cost required by singular value decompositions (SVD) which becomes increasingly expensive as the sizes and ranks of the underlying matrices increase. It is therefore desirable to exploit an alternative approach more suitable for solving large-scale problems.

In this paper, we investigate solving a more explicit model other than minimizing the nuclear norm in (1.2), thus avoiding SVD computation all together. Our goal is simply finding a low-rank matrix W so that $\|\mathcal{P}_\Omega(W - M)\|_F^2$ or equivalently the distance between W and the set $\{Z \in \mathbb{R}^{m \times n}, Z_{ij} = M_{ij}, \forall (i, j) \in \Omega\}$ is minimized. Obviously, any matrix $W \in \mathbb{R}^{m \times n}$ having a rank up to K can be expressed as a matrix multiplication $W = XY$ where $X \in \mathbb{R}^{m \times K}$ and $Y \in \mathbb{R}^{K \times n}$. Now we propose solving the following non-convex relaxation of (1.1),

$$(1.4) \qquad \min_{X,Y,Z} \frac{1}{2}\|XY - Z\|_F^2 \text{ s.t. } Z_{ij} = M_{ij}, \forall (i, j) \in \Omega,$$

where $X \in \mathbb{R}^{m \times K}, Y \in \mathbb{R}^{K \times n}, Z \in \mathbb{R}^{m \times n}$, and K will be dynamically adjusted. The premise of introducing the low-rank factorization model (1.4) is that hopefully it is much faster to solve this model than model (1.2). However, there are two potential drawbacks of the low-rank factorization model (1.4): (a) the non-convexity in the model may prevent one from getting a global solution, and (b) the approach requires an initial rank estimate K. In this paper, we present convincing evidences to show that (a) on a wide range of problems tested, the low-rank factorization model (1.4) is empirically as reliable as the nuclear norm minimization model (1.2); and (b) the initial rank estimate need not be close to the exact rank r of M (though one can benefit computationally from a good estimate). For example, we allow a strategy of starting from $K = 1$ and gradually increasing K. It is easy to observe that the global optimal value of (1.4) is monotonically non-increasing with respect to K. In principle, if K is smaller than the unknown rank r, the quality of the solution in terms of the objective function value can be improved by minimizing (1.4) again with an appropriately increased rank estimate. We mention that the introduction of the (splitting) variable Z is for a computational purpose that should become clear later.

A recent work in [14] is also based on a low-rank factorization model closely related to (1.4) where the factorization is in the form of USV^T where U and V have orthonormal columns. The authors derived a theoretical guarantee of recovery with high probability for their approach that consists of three steps. The first step is called trimming that removes from the sample $\mathcal{P}_\Omega(M)$ "over-represented" rows or columns. The second step finds the best rank-r approximation matrix to the remaining sample matrix via singular value decomposition (SVD) where r is the true rank and assumed to be known. In the final step, starting from the computed SVD factor as an initial guess, they solve the factorization model via a special gradient descent method that keeps the variables U and V orthonormal. The key intuition for their theoretical result is that the initial guess is so good that it falls into a certain neighborhood of the global minimum where there exists no other stationary point with high probability. This enables the authors to prove that their gradient descent method generates a sequence residing within this small neighborhood and converging to the global solution in the limit, despite the non-convexity of the factorization model. Given that our factorization model (1.4) is essentially the same as theirs, our approach should be able to benefit from the same initial point and possibly attain a similar theoretical guarantee. However, the proofs in [14] are specially tailored to the particularities of their algorithm and do not apply to our algorithm presented in this paper. Extending a similar theoretical result to our case is a topic of interest for future research. Meanwhile, the present paper concentrates on algorithm construction,

convergence (to stationary point) analysis and performance evaluations.

Our main contribution is the development of an efficient algorithm for (1.4) that can reliably solve a wide range of matrix completion and approximation problems at a speed much faster than the best of existing nuclear norm minimization algorithms. Like in many other similar cases, the structure of (1.4) suggests an alternating minimization scheme. In this case, one can update each of the variables X, Y or Z efficiently while fixing the other two. The subproblems with respect to either the variable X or Y are linear least squares problems only involving $K \times K$ coefficient matrices in their normal equations, and the solution of the subproblem for Z only needs the product XY. This alternating minimization procedure is also called a nonlinear (block) Gauss-Seidel (GS) scheme or a block coordinate descent method. In this paper, we propose a more sophisticated nonlinear successive over-relaxation (SOR) scheme with a strategy to adjust the relaxation weight dynamically. Numerical experiments show that this new scheme is significantly faster than the straightforward nonlinear GS scheme. Our approaches can also be applied to problems with general linear constraints. The convergence of nonlinear GS (coordinate descent) methods for several optimization problems has been studied, for example, in [31, 21, 10, 32]. However, we are unaware of any general convergence result for nonlinear SOR methods on non-convex optimization that is directly applicable to our nonlinear SOR algorithm. In this paper, we proved that our approach converges to a stationary point under a very mild assumption.

The rest of this paper is organized as follows. We first present an alternating minimization scheme for (1.4) in section 2.1 with two efficient implementation variants. Our nonlinear SOR algorithm is introduced in section 2.2 and an extension to general linearly constrained problems is briefly discussed in section 2.3. An convergence analysis for the nonlinear SOR algorithm is given in section 3. Finally, two strategies for adjusting the rank estimate K and numerical results are presented in section 4 to demonstrate the robustness and efficiency of our algorithm.

2. Alternating minimization schemes.

2.1. Nonlinear Gauss-Seidal method.
We start with a straightforward alternating minimization scheme for solving problem (1.4). Although alternating minimization is a common strategy widely used in many other similar situations, there is a subtlety in this case in terms of efficiency of implementation. Given the current iterates X, Y and Z, the algorithm updates these three variables by minimizing (1.4) with respect to each one separately while fixing the other two. For example, by fixing the values of Y and Z, we obtain the new point X_+:

$$X_+ = ZY^\top(YY^\top)^\dagger = \operatorname*{argmin}_{X \in \mathbb{R}^{m \times K}} \frac{1}{2}\|XY - Z\|_F^2,$$

where the Moore-Penrose pseudo-inverse matrix $A^\dagger$ of a given matrix A is a generalization of the inverse matrix and defined as the unique matrix satisfying $AA^\dagger A = A$, $A^\dagger AA^\dagger = A^\dagger$, $(AA^\dagger)^\top = AA^\dagger$ and $(A^\dagger A)^\top = A^\dagger A$. Similarly, we can compute Y_+ and Z_+ using the latest available values for fixed variables. This procedure yields the following iterative scheme:

(2.1a) $$X_+ \leftarrow ZY^\dagger \equiv ZY^\top(YY^\top)^\dagger,$$

(2.1b) $$Y_+ \leftarrow (X_+)^\dagger Z \equiv (X_+^\top X_+)^\dagger(X_+^\top Z),$$

(2.1c) $$Z_+ \leftarrow X_+Y_+ + \mathcal{P}_\Omega(M - X_+Y_+).$$

The only $m \times n$ matrix involved is Z_+ which is the sum of a low rank product X_+Y_+ and a sparse matrix $\mathcal{P}_\Omega(M - X_+Y_+)$.

It follows from (2.1a) and (2.1b) that

$$X_+Y_+ = \left(X_+(X_+^\top X_+)^\dagger X_+^\top\right) Z = \mathcal{P}_{X_+} Z,$$

where $\mathcal{P}_A := A(A^\top A)^\dagger A^\top = QQ^\top$ is the orthogonal projection onto the range space $\mathcal{R}(A)$ of A and $Q := \mathrm{orth}(A)$ is an orthonormal basis for $\mathcal{R}(A)$. The pseudo-inverse of A, the orthonormal basis of $\mathcal{R}(A)$ and the orthogonal projection onto $\mathcal{R}(A)$ can be computed from either the SVD or the QR factorization of A. One can verify that $\mathcal{R}(X_+) = \mathcal{R}(ZY^\top)$. Indeed, let $Y = U\Sigma V^\top$ be the SVD of Y, then $X_+ = ZV\Sigma^\dagger U^\top$ and $ZY^\top = ZV\Sigma U^\top$, implying that $\mathcal{R}(X_+) = \mathcal{R}(ZY^\top) = \mathcal{R}(ZV)$ and leading to the following lemma.

LEMMA 2.1. *Let* (X_+, Y_+) *be generated by* (2.1). *There holds*

$$(2.2) \qquad X_+ Y_+ = \mathcal{P}_{ZY^\top} Z = ZY^\top (YZ^\top ZY^\top)^\dagger (YZ^\top)Z.$$

We next present two iterative schemes equivalent to (2.1). Since the objective function (1.4) is determined by the product $X_+ Y_+$, different values of X_+ and Y_+ are essentially equivalent as long as they give the same product $X_+ Y_+$. Lemma 2.1 shows that the inversion $(YY^\top)^\dagger$ can be saved when the projection $\mathcal{P}_{ZY^\top}$ is computed. The unique feature of our new schemes is that only one least square problem is involved at each iteration. The first variant is to replace the step (2.1a) by

$$(2.3a) \qquad X_+ \leftarrow ZY^\top,$$

while Y_+ and Z_+ are still generated by step (2.1b) and (2.1c). The second variant computes the orthogonal projection $\mathcal{P}_{ZY^\top} = VV^\top$, where $V := \mathrm{orth}(ZY^\top)$ is an orthogonal basis of $\mathcal{R}(ZY^\top)$. Hence, (2.2) can be rewritten as $X_+ Y_+ = VV^\top Z$ and one can derive:

$$(2.4a) \qquad X_+ \leftarrow V,$$
$$(2.4b) \qquad Y_+ \leftarrow V^\top Z,$$

while Z_+ is still generated by step (2.1c). The scheme (2.4) is often preferred since computing the step (2.4b) by QR factorization is generally more stable than solving the normal equations. Note that the schemes (2.1), (2.3) and (2.4) can be used interchangeably in deriving properties of $X_+ Y_+$.

Our computational experience indicates that although the above alternating minimization approach appears to be reliable for matrix completion, its convergence rate can be slow in recovering large-sized matrices with very low ranks. A possible acceleration technique may involve applying an extension of the classic augmented-Lagrangian-based alternating direction method (ADM) for convex optimization to the factorization model (see [36] for such an ADM extension). However, in this paper, we investigate a nonlinear Successive Over-Relaxation (SOR) approach that we found to be particularly effective for solving the matrix completion problem.

2.2. A Nonlinear SOR-like Scheme. In this section, we present a nonlinear Successive Over-Relaxation (SOR) approach to accelerate convergence. By introducing a Lagrange multiplier $\Lambda \in \mathbb{R}^{m \times n}$ so that $\Lambda = \mathcal{P}_\Omega(\Lambda)$, the Lagrangian function of (1.4) is defined as

$$(2.5) \qquad \mathcal{L}(X, Y, Z, \Lambda) = \frac{1}{2}\|XY - Z\|_F^2 - \Lambda \bullet \mathcal{P}_\Omega(Z - M),$$

where the inner product between two matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{m \times n}$ is defined as $A \bullet B := \sum_{jk} A_{j,k} B_{j,k}$. Differentiating the Lagrangian function $\mathcal{L}(X, Y, Z, \Lambda)$, we have the first-order optimality conditions for (1.4):

$$(2.6a) \qquad\qquad (XY - Z)Y^\top = 0,$$

$$(2.6b) \qquad\qquad X^\top(XY - Z) = 0,$$

$$(2.6c) \qquad\qquad \mathcal{P}_{\Omega^c}(Z - XY) = 0,$$

$$(2.6d) \qquad\qquad \mathcal{P}_{\Omega}(Z - M) = 0,$$

plus the equations

$$(2.7) \qquad\qquad \mathcal{P}_{\Omega}(Z - XY) = \Lambda.$$

Clearly, the multiplier matrix Λ measures the residual $Z - XY$ in Ω and has no effect in the process of determining X, Y, Z. It is also easy to see that the alternating minimization schemes in section 2.1 are exactly a Gauss-Seidal (GS) method applied to the nonlinear and square system (2.6).

In numerical linear algebra, the method of successive over-relaxation (SOR) [9] for solving a linear system of equations is devised by applying extrapolation to the GS method, that is, the new trial point is a weighted average between the previous iterate and the computed GS iterate successively for each component. A proper value of the weight often results in faster convergence. Applying the same idea to the basic schemes (2.1), (2.3) and (2.4) gives a nonlinear SOR scheme:

$$(2.8a) \qquad\qquad X_+ \leftarrow ZY^\top(YY^\top)^\dagger,$$

$$(2.8b) \qquad\qquad X_+(\omega) \leftarrow \omega X_+ + (1 - \omega)X,$$

$$(2.8c) \qquad\qquad Y_+ \leftarrow (X_+(\omega)^\top X_+(\omega))^\dagger(X_+(\omega)^\top Z),$$

$$(2.8d) \qquad\qquad Y_+(\omega) \leftarrow \omega Y_+ + (1 - \omega)Y,$$

$$(2.8e) \qquad\qquad Z_+(\omega) \leftarrow X_+(\omega)Y_+(\omega) + \mathcal{P}_{\Omega}(M - X_+(\omega)Y_+(\omega)),$$

where the weight $\omega \geq 1$. Obviously, $\omega = 1$ gives the GS method.

Assuming that the matrix Y has full row rank, the two least squares problems in (2.8) can be reduced into one like the second basic scheme (2.3). Let us denote the residual by

$$(2.9) \qquad\qquad S = \mathcal{P}_{\Omega}(M - XY),$$

which will be used to measure optimality. After each iteration, the variable Z, which is feasible, can be expressed as $Z = XY + S$. Let Z_ω be a weighted sum of the matrices XY and S, that is,

$$(2.10) \qquad\qquad Z_\omega \triangleq XY + \omega S = \omega Z + (1 - \omega)XY.$$

Using the fact that the matrix $YY^\top(YY^\top)^\dagger$ is the identity from our assumption, we obtain

$$Z_\omega Y^\top(YY^\top)^\dagger = \omega ZY^\top(YY^\top)^\dagger + (1 - \omega)XYY^\top(YY^\top)^\dagger$$
$$= \omega X_+ + (1 - \omega)X,$$

which is exactly the step (2.8b). The Y steps can be verified in a similar fashion. Therefore, the SOR algorithm (2.8)

is equivalent to the following scheme:

$$(2.11a) \qquad X_+(\omega) \leftarrow Z_\omega Y^\top \text{ or } Z_\omega Y^\top (YY^\top)^\dagger,$$

$$(2.11b) \qquad Y_+(\omega) \leftarrow (X_+(\omega)^\top X_+(\omega))^\dagger (X_+(\omega)^\top Z_\omega),$$

$$(2.11c) \qquad \mathcal{P}_{\Omega^c}(Z_+(\omega)) \leftarrow \mathcal{P}_{\Omega^c}(X_+(\omega)Y_+(\omega)),$$

$$(2.11d) \qquad \mathcal{P}_\Omega(Z_+(\omega)) \leftarrow \mathcal{P}_\Omega(M).$$

Again, an implementation with a single QR decomposition can be utilized just as in scheme (2.4).

We next present a strategy for choosing suitable weight values ω so that scheme (2.11) can converge. Certainly, a fixed weight is not good enough for nonlinear and non-convex functions. Our strategy is to adjust the weight ω dynamically according to the change of the objective function values. After the point $(X_+(\omega), Y_+(\omega), Z_+(\omega))$ is computed, we calculate the residual ratio

$$(2.12) \qquad \gamma(\omega) = \frac{\|S_+(\omega)\|_F}{\|S\|_F},$$

where

$$(2.13) \qquad S_+(\omega) \triangleq \mathcal{P}_\Omega(M - X_+(\omega)Y_+(\omega)).$$

If $\gamma(\omega) < 1$, this new pair of point is accepted and the step is called "succssful"; otherwise, the step is "unsuccessful", ω is reset to 1 and this procedure is repeated. Note that a small $\gamma(\omega)$ indicates that the current weight value ω works well so far. Hence, ω is increased only if the calculated point is acceptable but the residual ratio $\gamma(\omega)$ is considered "too large"; that is, $\gamma(\omega) \in [\gamma_1, 1)$ for some $\gamma_1 \in (0, 1)$. From the above considerations, we arrive at Algorithm 1 below.

Algorithm 1: A low-rank matrix fitting algorithm (`LMaFit`)

---

1 Input index set Ω, data $\mathcal{P}_\Omega(M)$ and a rank overestimate $K \geq r$.
2 Set $Y^0 \in \mathbb{R}^{K \times n}$, $Z^0 = \mathcal{P}_\Omega(M)$, $\omega = 1$, $\tilde{\omega} > 1$, $\delta > 0$, $\gamma_1 \in (0, 1)$ and $k = 0$.
3 **while** *not convergent* **do**
4 $\quad$ Compute $(X_+(\omega), Y_+(\omega), Z_+(\omega))$ according to (2.11) with $(X, Y, Z) = (X^k, Y^k, Z^k)$.
5 $\quad$ Compute the residual ratio $\gamma(\omega)$ according to (2.12).
6 $\quad$ **if** $\gamma(\omega) \geq 1$ **then** set $\omega = 1$ and go to step 4.
7 $\quad$ Update $(X^{k+1}, Y^{k+1}, Z^{k+1}) = (X_+(\omega), Y_+(\omega), Z_+(\omega))$ and increment k.
8 $\quad$ **if** $\gamma(\omega) \geq \gamma_1$ **then** set $\delta = \max(\delta, 0.25(\omega - 1))$ and $\omega = \min(\omega + \delta, \tilde{\omega})$.

For illustration, we compare the efficiency of the GS scheme (2.1) and the nonlinear SOR scheme (2.11) on two random matrices M with $m = n = 1000$, $r = 10$ with two different sampling ratios at, respectively, 0.08 and 0.15 (see subsection 4.2 for detailed construction procedure and algorithmic parameter setting). The algorithms were run by using two different rank estimations $K = 12$ and 20. The normalized residuals $\|\mathcal{P}_\Omega(M - XY)\|_F / \|\mathcal{P}_\Omega(M)\|_F$ are depicted in Figures 2.1 (a) and (b), respectively. The apparent jumps in the residuals were due to adjustments of rank estimations, which will be explained later. From the figures, it is evident that the nonlinear SOR scheme is significant faster than the nonlinear GS scheme.

FIG. 2.1. *Comparison between the nonlinear GS and SOR schemes*

(a) n=1000, r=10, sampling ratio = 0.08

(b) n=1000, r=10, sampling ratio=0.15
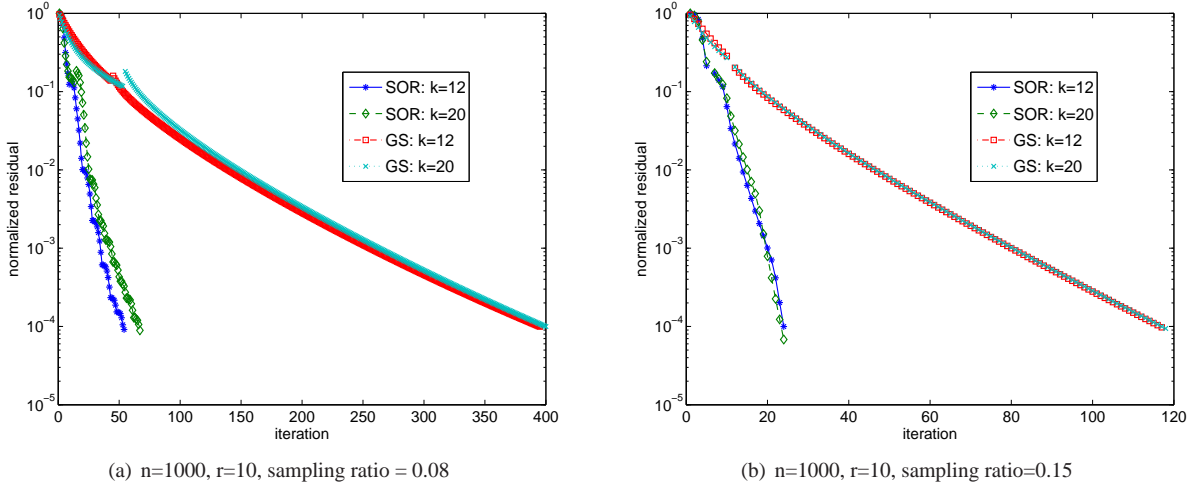
2.3. Extension to problems with general linear constraints. Both our nonlinear GS and SOR approaches can be applied to solving rank minimization problems with general linear constraints:

$$(2.14) \qquad \min_{W \in \mathbb{R}^{m \times n}} \text{rank}(W), \text{ s.t. } \mathcal{A}(W) = b,$$

where the linear operator $\mathcal{A} : \mathbb{R}^{m \times n} \to \mathbb{R}^q$ and the vector $b \in \mathbb{R}^q$ are given. Similar to (1.4), our nonconvex relaxation becomes

$$(2.15) \qquad \min_{X,Y,Z} \frac{1}{2} \|XY - Z\|_F^2 \text{ s.t. } \mathcal{A}(Z) = b,$$

where $X \in \mathbb{R}^{m \times K}, Y \in \mathbb{R}^{K \times n}, Z \in \mathbb{R}^{m \times n}$. For any fixed X and Y, the minimizer Z_+ of (2.15) with respect to Z is

$$Z_+ = XY + \mathcal{A}^\top (\mathcal{A}\mathcal{A}^\top)^\dagger (b - \mathcal{A}(XY)).$$

Let $S := \mathcal{A}^\top (\mathcal{A}\mathcal{A}^\top)^\dagger (b - \mathcal{A}(XY))$ and $Z_\omega = XY + \omega S$. Similar to the scheme (2.11), we obtain

$$(2.16a) \qquad X_+(\omega) \leftarrow Z_\omega Y^\top \text{ or } Z_\omega Y^\top (YY^\top)^\dagger,$$

$$(2.16b) \qquad Y_+(\omega) \leftarrow (X_+(\omega)^\top X_+(\omega))^\dagger (X_+(\omega)^\top Z_\omega),$$

$$(2.16c) \qquad Z_+(\omega) \leftarrow X_+(\omega)Y_+(\omega) + \mathcal{A}^\top (\mathcal{A}\mathcal{A}^\top)^\dagger (b - \mathcal{A}(X_+(\omega)Y_+(\omega))).$$

3. Convergence Analysis. We now analyze Algorithm 1 by revealing the relationships between the residuals $\|S\|_F$ and $\|S_+(\omega)\|_F$. Let $V(\omega) := \texttt{orth}(X_+(\omega))$ and $U := \texttt{orth}(Y^\top)$ be orthogonal bases of the range spaces of $\mathcal{R}(X_+(\omega))$ and $\mathcal{R}(Y^\top)$, respectively. Consequently, the orthogonal projections onto $\mathcal{R}(X_+(\omega))$ and $\mathcal{R}(Y^\top)$ can be expressed as:

$$Q(\omega) := V(\omega)V(\omega)^\top = X_+(\omega)(X_+(\omega)^\top X_+(\omega))^\dagger X_+(\omega)^\top,$$
$$P := UU^\top = Y^\top (YY^\top)^\dagger Y.$$

We list several useful identities that can be verified from the definition of pseudo-inverse. For any $A \in \mathbb{R}^{m \times n}$,

$$
\begin{aligned}
A^\dagger &= A^\dagger (A^\dagger)^\top A^\top = A^\top (A^\dagger)^\top A^\dagger = (A^\top A)^\dagger A^\top = A^\top (AA^\top)^\dagger, \\
A &= (A^\dagger)^\top A^\top A = AA^\top (A^\dagger)^\top.
\end{aligned}
$$

(3.1)

The lemma below and its proof will provide us a key equality.

LEMMA 3.1. *Let $(X_+(\omega), Y_+(\omega))$ be generated by (2.11). There holds*

(3.2)
$$
\omega S \bullet (X_+(\omega)Y_+(\omega) - XY) = \|X_+(\omega)Y_+(\omega) - XY\|_F^2.
$$

Proof. It follows from $Y^\top = Y^\dagger YY^\top$ (see (3.1)), $X_+(\omega) = Z_\omega Y^\dagger$ and $Z_\omega = XY + \omega S$ that

$$
X_+(\omega)YY^\top = Z_\omega Y^\top = XYY^\top + \omega SY^\top.
$$

Post-multiplying both sides by $(YY^\top)^\dagger Y$ and rearranging, we have $(X_+(\omega) - X)Y = \omega SY^\top (YY^\top)^\dagger Y$; i.e.,

(3.3)
$$
(X_+(\omega) - X)Y = \omega SP.
$$

On the other hand, the equalities $X_+(\omega)^\top = X_+(\omega)^\top X_+(\omega)(X_+(\omega))^\dagger$ and (3.3) yield

$$
\begin{aligned}
X_+(\omega)^\top X_+(\omega)Y_+(\omega) &= X_+(\omega)^\top Z_\omega = X_+(\omega)^\top (XY + \omega S) \\
&= X_+(\omega)^\top (X_+(\omega)Y - (X_+(\omega) - X)Y + \omega S) \\
&= X_+(\omega)^\top X_+(\omega)Y + \omega X_+(\omega)^\top S(I - P).
\end{aligned}
$$

Pre-multiplying both sided by $X_+(\omega)(X_+(\omega)^\top X_+(\omega))^\dagger$ and rearranging, we arrive at

(3.4)
$$
X_+(\omega)(Y_+(\omega) - Y) = \omega Q(\omega)S(I - P).
$$

Therefore, in view of (3.3) and (3.4), we obtain

$$
\begin{aligned}
X_+(\omega)Y_+(\omega) - XY &= (X_+(\omega) - X)Y + X_+(\omega)(Y_+(\omega) - Y) \\
&= \omega SP + \omega Q(\omega)S(I - P) \\
&= \omega(I - Q(\omega))SP + \omega Q(\omega)S.
\end{aligned}
$$

(3.5)

(3.6)

Therefore,

(3.7)
$$
\|X_+(\omega)Y_+(\omega) - XY\|_F^2 = \omega^2 \|(I - Q(\omega))SP\|_F^2 + \omega^2 \|Q(\omega)S\|_F^2.
$$

Finally, in view of (3.6) and the properties of orthogonal projections, we have:

$$
\begin{aligned}
\omega S \bullet (X_+(\omega)Y_+(\omega) - XY) &= \omega^2 S \bullet (I - Q(\omega))SP + \omega^2 S \bullet Q(\omega)S \\
&= \omega^2 SP \bullet (I - Q(\omega))SP + \omega^2 S \bullet Q(\omega)S \\
&= \omega^2 \|(I - Q(\omega))SP\|_F^2 + \omega^2 \|Q(\omega)S\|_F^2 \\
&= \|X_+(\omega)Y_+(\omega) - XY\|_F^2,
\end{aligned}
$$

8

which proves the lemma. □

It is easy to see that

(3.8)
$$\frac{1}{\omega}\|XY - Z_\omega\|_F = \|S\|_F.$$

Therefore, after the first two steps in (2.11),

$$\frac{1}{\omega}\|X_+(\omega)Y_+(\omega) - Z_\omega\|_F \leq \|S\|_F$$

and the strict inequality holds unless that the first two equations of the optimality conditions of (2.6) already hold. Or equivalently,

(3.9)
$$\frac{1}{\omega^2}\left(\|\mathcal{P}_{\Omega^c}(X_+(\omega)Y_+(\omega) - Z_\omega)\|_F^2 + \|\mathcal{P}_{\Omega}(X_+(\omega)Y_+(\omega) - Z_\omega)\|_F^2\right) \leq \|S\|_F^2.$$

LEMMA 3.2. *Let* $(X_+(\omega), Y_+(\omega))$ *be generated by (2.11) for any* $\omega \geq 1$, *then*

(3.10)
$$\frac{1}{\omega^2}\|X_+(\omega)Y_+(\omega) - Z_\omega\|_F^2 = \|(I - Q(\omega))S(I - P)\|_F^2 = \|S\|_F^2 - \rho_{12}(\omega),$$

where

(3.11)
$$\rho_{12}(\omega) \triangleq \|SP\|_F^2 + \|Q(\omega)S(I - P)\|_F^2$$

is the amount of residual reduction from $\|S\|_F^2$ *after the first two steps of (2.11).*

Proof. From the definition of Z_ω and (3.5), we obtain

$$X_+(\omega)Y_+(\omega) - Z_\omega = X_+(\omega)Y_+(\omega) - XY - \omega S = \omega SP + \omega Q(\omega)S(I - P) - \omega S$$
$$= -\omega(I - Q(\omega))S(I - P),$$

which proves the right equality in (3.10). Using (3.2) and (3.7), we have:

$$\|X_+(\omega)Y_+(\omega) - Z_\omega\|_F^2 = \|X_+(\omega)Y_+(\omega) - XY\|_F^2 + \omega^2\|S\|_F^2 - 2\omega S \bullet (X_+(\omega)Y_+(\omega) - XY)$$
$$= \omega^2\|S\|_F^2 - \|X_+(\omega)Y_+(\omega) - XY\|_F^2$$
$$= \omega^2\|S\|_F^2 - \omega^2\rho_{12}(\omega),$$

which proves the second equality in (3.10). □

After the third step in (2.11), we have

$$\|\mathcal{P}_{\Omega^c}(X_+(\omega)Y_+(\omega) - Z_+(\omega))\|_F = 0,$$

since $\mathcal{P}_{\Omega^c}(Z_\omega) \equiv \mathcal{P}_{\Omega^c}(XY)$ independent of ω. Consequently, the residual reduction after the third step is

(3.12)
$$\rho_3(\omega) \triangleq \frac{1}{\omega^2}\|\mathcal{P}_{\Omega^c}(X_+(\omega)Y_+(\omega) - XY)\|_F^2.$$

9

Another formula for ρ_3 is

$$(3.13) \qquad \rho_3(\omega) \triangleq \|\mathcal{P}_{\Omega^c}(SP + Q(\omega)S(I - P))\|_F^2,$$

which follows from the equality (3.2).

The change of the residual value after the fourth step is

$$\rho_4(\omega) \triangleq \frac{1}{\omega^2}\|\mathcal{P}_\Omega(X_+(\omega)Y_+(\omega) - Z_\omega)\|_F^2 - \|S_+(\omega)\|_F^2;$$

or equivalently,

$$(3.14) \qquad \rho_4(\omega) \triangleq \frac{1}{\omega^2}\|S_+(\omega) + (\omega - 1)S\|_F^2 - \|S_+(\omega)\|_F^2 \geq 0.$$

Clearly, $\rho_4(1) = 0$. For $\omega > 1$, it follows from (3.14) that

$$(3.15) \qquad \frac{\omega^2 \rho_4(\omega)}{\omega - 1} = (\omega - 1)(\|S\|_F^2 - \|S_+(\omega)\|_F^2) - 2S_+(\omega) \bullet (S_+(\omega) - S).$$

We will show next that the rate of change of $\rho_4(\omega)$ at $\omega = 1^+$ is nonnegative.

Lemma 3.3.

$$(3.16) \qquad \lim_{\omega \to 1^+} \frac{\rho_4(\omega)}{\omega - 1} = 2\|\mathcal{P}_{\Omega^c}(X_+(1)Y_+(1) - XY)\|_F^2 \geq 0.$$

Proof. Let $\omega \to 1$ and $S_+ \triangleq S_+(1)$. We obtain from the definitions of S and S_+, and (3.2) that

$$\begin{aligned}
\lim_{\omega \to 1^+} \frac{\rho_4(\omega)}{\omega - 1} = \lim_{\omega \to 1^+} \frac{\omega^2 \rho_4(\omega)}{\omega - 1} &= -2S_+ \bullet (S_+ - S) \\
&= -2\|S_+ - S\|_F^2 - 2S \bullet (S_+ - S) \\
&= -2\|\mathcal{P}_\Omega(X_+Y_+ - XY)\|_F^2 + 2S \bullet (X_+Y_+ - XY) \\
&= 2\|\mathcal{P}_{\Omega^c}(X_+Y_+ - XY)\|_F^2,
\end{aligned}$$
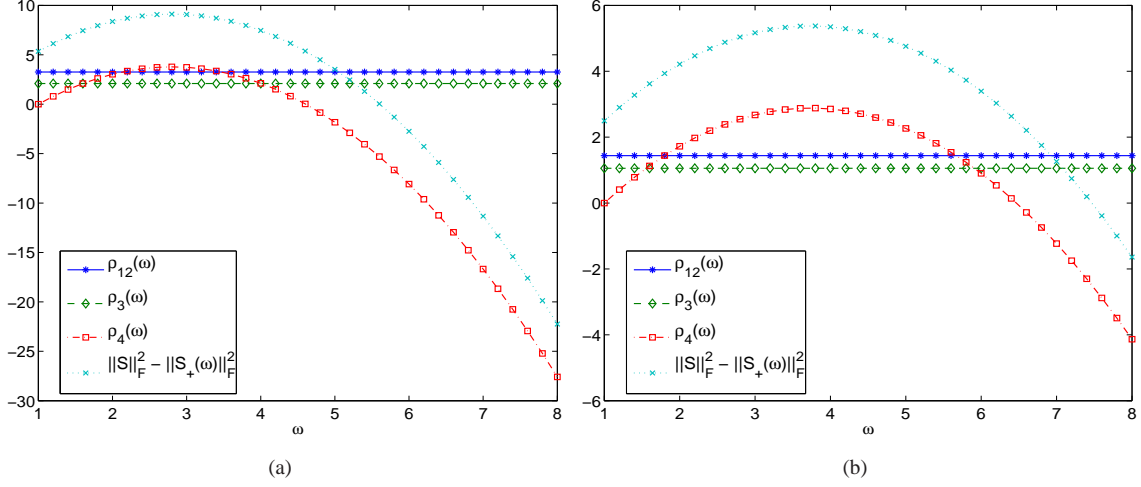
which completes the proof. $\square$

If $\rho_4(\omega)$ is continuous, then Lemma 3.3 guarantees that $\rho_4(\omega) > 0$ in some range of $\omega > 1$. In fact, suppose that $\text{rank}(Z_\omega) = \text{rank}(Z)$ as $\omega \to 1^+$. The equality $\text{rank}(YZ_\omega^\top Z_\omega Y^\top) = \text{rank}(YZ^\top ZY^\top)$ holds as $\omega \to 1^+$, hence, $\lim_{\omega \to 1^+}(YZ_\omega^\top Z_\omega Y^\top)^\dagger = (YZ^\top ZY^\top)^\dagger$ holds by [28]. The continuity of the product $X_+(\omega)Y_+(\omega)$ implies that $\rho_4(\omega)$ is continuous as $\omega \to 1^+$. For example, we depict the continuity of the functions $\rho_{12}(\omega)$, $\rho_3(\omega)$ on a randomly generated problem from two different pair of points (X, Y, Z) in Figures 3.1 (a) and (b). The benefit of increasing ω can be quite significant at times. For example, as can be seen from Figure 3.1 (b), when ω is increased from 1 to 4, the amount of total residual reduction is more than doubled.

We have proved the following result about the residual-reduction property of the nonlinear SOR algorithm.

THEOREM 3.4. *Assume that* $\text{rank}(Z_\omega) = \text{rank}(Z), \forall \omega \in [1, \omega_1]$ *for some* $\omega_1 \geq 1$. *Let* $(X_+(\omega), Y_+(\omega), Z_+(\omega))$ *be generated by the SOR scheme (2.11) starting from a non-stationary point* (X, Y, Z), *and* ρ_{12}, ρ_3 *and* ρ_4 *be defined as in (3.11), (3.12) and (3.14), respectively. Then there exists some* $\omega_2 \geq 1$ *such that*

$$(3.17) \qquad \|S\|_F^2 - \|S_+(\omega)\|_F^2 = \rho_{12}(\omega) + \rho_3(\omega) + \rho_4(\omega) > 0, \quad \forall \omega \in [1, \omega_2],$$

(a)   (b)

where $\rho_{12}(\omega), \rho_3(\omega) \geq 0$ by definition. Moreover, whenever $\rho_3(1) > 0$ (equivalently $\mathcal{P}_{\Omega^c}(X_+(1)Y_+(1) - XY) \neq 0$) and $\omega_1 > 1$, then $\omega_2 > 1$ can be chosen so that $\rho_4(\omega) > 0, \forall \omega \in (1, \omega_2]$.

Next we present a convergence result for our algorithm. Since model (1.4) is non-convex, we are only able to establish convergence to a stationary point under a mild assumption. Note that the objective function is bounded below by zero and is decreased by at least an amount of ρ_3 at every iteration. There must hold (see (3.12))

$$\mathcal{P}_{\Omega^c}(X^{k+1}Y^{k+1} - X^k Y^k) \to 0.$$

In light of the above, it is entirely reasonable to assume that $\{\mathcal{P}_{\Omega^c}(X^k Y^k)\}$ remains bounded, barring the unlikely alternative that $\|\mathcal{P}_{\Omega^c}(X^k Y^k)\| \to \infty$.

THEOREM 3.5. *Let $\{(X^k, Y^k, Z^k)\}$ be generated by Algorithm 1 and $\{\mathcal{P}_{\Omega^c}(X^k Y^k)\}$ be bounded. Then there exists at least a subsequence of $\{(X^k, Y^k, Z^k)\}$ that satisfies the first-order optimality conditions (2.6) in the limit.*

Proof. It follows from the boundedness of $\{\mathcal{P}_{\Omega^c}(X^k Y^k)\}$ and the algorithm construction that both $\{Z^k\}$ and $\{X^k Y^k\}$ are bounded sequences. It suffices to prove (2.6a)-(2.6b) since the other conditions are satisfied by the construction of Algorithm 1. Without loss of generality, we assume that $\{X^k\}$ is generated by a scheme analogous to (2.4): given $(X, Y) = (X^k, Y^k)$ and $\omega \in [1, \tilde{\omega}]$

$$Z_\omega = \omega Z + (1 - \omega)XY, \quad X_+ = \mathrm{orth}(Z_\omega Y^T), \quad Y_+ = X_+^T Z_w.$$

Obviously, $\{X^k\}$ is bounded. In addition, $\{Y^k\}$ is also bounded due to the boundedness of both $\{Z^k\}$ and $\{X^k Y^k\}$.

Let $\mathcal{I} = \{k : \ \rho_4^k(\omega^k) \geq 0\}$, and $\mathcal{I}^c$ be the complement of $\mathcal{I}$. It follows from (3.17) that

$$(3.18) \qquad \|S^0\|_F^2 \geq \sum_{i \in \mathcal{I}} \rho_{12}^i(\omega) = \sum_{i \in \mathcal{I}} \|S^i P^i\|_F^2 + \|Q^i S^i (I - P^i)\|_F^2.$$

We consider the following three cases.

i) Suppose $|\mathcal{I}^c| < \infty$. It follows from (3.18) that

$$(3.19) \qquad \lim_{i \to \infty} \|S^i P^i\|_F^2 = 0 \text{ and } \lim_{i \to \infty} \|Q^i S^i\|_F^2 = 0.$$

11

The construction of the scheme (2.11) gives the equalities:

$$\mathcal{P}_\Omega(M) = \mathcal{P}_\Omega(Z^i), \quad \mathcal{P}_{\Omega^c}(Z^i) = \mathcal{P}_{\Omega^c}(X^i Y^i), \quad P^i = U^i(U^i)^\top,$$

where $U^i = \texttt{orth}((Y^i)^\top)$. Therefore, we obtain

$$S^i P^i = \mathcal{P}_\Omega(M - X^i Y^i) P^i = \mathcal{P}_\Omega(Z^i - X^i Y^i) P^i = (Z^i - X^i Y^i) P^i = (Z^i - X^i Y^i) U^i(U^i)^\top,$$

which yields $\lim_{i \to \infty}(Z^i - X^i Y^i) U^i = 0$ in view of the first part of (3.19). Since U^i is an orthonormal basis for $\mathcal{R}((Y^i)^\top)$ and the sequence $\{Y^i\}$ is bounded, we have

(3.20)
$$\lim_{i \to \infty}(Z^i - X^i Y^i)(Y^i)^\top = 0.$$

Using $Q^i = V^i(V^i)^\top$, where V^i is an orthonormal basis for $\mathcal{R}(X^{i+1})$, we obtain

(3.21)
$$Q^i S^i = Q^i S^{i+1} + Q^i(S^i - S^{i+1}) = V^i(V^i)^\top(Z^{i+1} - X^{i+1} Y^{i+1}) + Q^i(S^i - S^{i+1}).$$

Using (3.7) and (3.19), we obtain

$$\|S^i - S^{i+1}\|_F^2 \leq \|X^{i+1} Y^{i+1} - X^i Y^i\|_F^2 \leq (\tilde{\omega})^2(\|S^i P^i\|_F^2 + \|Q^i S^i(I - P^i)\|_F^2) \to 0,$$

hence, $\lim_{i \to \infty} \|S^i - S^{i+1}\|_F = 0$. This fact, together with (3.19) and (3.21), proves

$$(V^i)^\top(Z^{i+1} - X^{i+1} Y^{i+1}) \to 0.$$

In view of the boundedness of $\{X^i\}$, we arrive at

(3.22)
$$\lim_{i \to \infty}(X^i)^\top(X^i Y^i - Z^i) = 0.$$

ii) Suppose $|\mathcal{I}^c| = \infty$ and $|\{k \in \mathcal{I}^c : \gamma(\omega^k) \geq \gamma_1\}| < \infty$. That is, for $k \in \mathcal{I}^c$ sufficiently large we have

$$\|S^{k+1}\|_F < \gamma_1 \|S^k\|_F.$$

Consquently, $\lim_{k \to \infty, k \in \mathcal{I}^c} \|S^k\|_F = 0$. Since $\|S^k\|$ is nonincreasing, the full sequence converges to the global minimizer of (1.4).

iii) Suppose $|\mathcal{I}^c| = \infty$ and $|\{i \in \mathcal{I}^c : \gamma(\omega^i) \geq \gamma_1\}| = \infty$. Then Algorithm 1 resets $\omega^i = 1$ for an infinite number of iterations. We obtain from (3.18) that

(3.23)
$$\|S^0\|_F^2 \geq \sum_{i \in \mathcal{I}_1} \rho_{12}^i(\omega) = \sum_{i \in \mathcal{I}_1} \|S^i P^i\|_F^2 + \|Q^i S^i(I - P^i)\|_F^2.$$

Hence, the subsequence in $\mathcal{I}_1$ satisfies (3.20) and (3.22) by repeating, in an analogous fashion, the proof of part i). $\square$

4. Computational Results. In this section, we report numerical results on our nonlinear SOR algorithm and other algorithms. The code `LMaFit` [35] for our algorithm is implemented in Matlab with a couple of small tasks written in C to avoid ineffective memory usage in Matlab. Other tested solvers include `APGL` [29], `FPCA` [22] and `OptSpace` [16], where the first two are nuclear minimization codes implemented under the Matlab environment.

`APGL` also utilizes a Matlab version (with the task of reorthogonalization implemented in C) of the SVD package PROPACK [17], and `FPCA` uses a fast Monte Carlo algorithm for SVD calculations implemented in Matlab. The code `OptSpace`, which has a Matlab version that was used in our tests, solves a model closely related to (1.4) using a gradient descent approach and starting from a specially constructed initial guess. All experiments were performed on a Lenovo D20 Workstation with two Intel Xeon E5506 Processors and 10GB of RAM.

We tested and compared these solvers on two classes of matrix problems: completion and low-rank approximation. The key difference between the two classes lies in whether a given sample is from a true low-rank matrix (with or without noise) or not. Although theoretical guarantees exist for matrix completion, to the best of our knowledge no such guarantees exist for low-rank approximation if samples are taken from a matrix of mathematically full rank. On the other hand, low-rank approximation problems are more likely to appear in practical applications.

4.1. Implementation details and rank estimation. Algorithm 1 starts from an initial guess $Y^0 \in \mathbb{R}^{K \times n}$. For the sake of simplicity, in all our experiments we set Y^0 to a diagonal matrix with 1's on the diagonal even though more elaborate choices certainly exist that may lead to better performance. The default values of the parameters $\tilde{\omega}$, δ and γ_1 were set to $+\infty$, 1 and 0.7, respectively. Since the increment δ is non-decreasing in Algorithm 1, the parameter ω can be increased too fast. Hence, we also reset δ to $0.1 * \max(\omega - 1, \delta)$ whenever $\gamma(\omega) \geq 1$. The stopping criteria in our numerical experiments follow

$$\texttt{relres} = \frac{\|\mathcal{P}_\Omega(M - X^kY^k)\|_F}{\|\mathcal{P}_\Omega(M)\|_F} \leq \texttt{tol} \text{ and } \texttt{reschg} = \left|1 - \frac{\|\mathcal{P}_\Omega(M - X^kY^k)\|_F}{\|\mathcal{P}_\Omega(M - X^{k-1}Y^{k-1})\|_F}\right| \leq \texttt{tol}/2,$$

where `tol` is a moderately small number.

Since a proper estimation to the rank K for the model (1.4) is essential for the success of `LMaFit`, two heuristic strategies for choosing K were implemented. In the first strategy, we start from a large K ($K \geq r$) and decrease it aggressively once a dramatic change in the estimated rank of the variable X is detected based on its QR factorization which usually occurs after a few iterations. Specifically, let $QR = XE$ be the economy-size QR factorization of X, where E is a permutation matrix so that $d := |\text{diag}(R)|$ is non-increasing, where $\text{diag}(R)$ is a vector whose ith component is R_{ii}. We compute the quotient sequnce $\tilde{d}_i = d_i/d_{i+1}$, $i = 1, \cdots, K - 1$, and examine the ratio

$$\tau = \frac{(K-1)\tilde{d}(p)}{\sum_{i \neq p}\tilde{d}_i},$$

where $\tilde{d}(p)$ is the maximal element of $\{\tilde{d}_i\}$ and p is the corresponding index. A large τ value indicates a large drop in the magnitude of d right after the p-th element. In the current implementation, we reset K to p once $\tau > 10$, and this adjustment is done only one time. On the other hand, by starting from a small initial guess, our second strategy is to increase K to $\min(K + \kappa, \texttt{rank\_max})$ when the algorithm stagnates, i.e., `reschg<10*tol`. Here, `rank_max` is the maximal rank estimation, and the increment $\kappa = \texttt{rk\_inc}$ if the current $K < 50$; otherwise, $\kappa = 2 * \texttt{rk\_inc}$. The default value of `rk_inc` is 5. In our code `LMaFit`, the first and second (or decreasing and increasing rank) strategies can be specified by setting the option `est_rank` to 1 or 2, respectively, and will be called the decreasing rank and increasing rank strategies, respectively.

Each strategy has its own advantages and disadvantages, and should be selected according to the properties of the targeted problems. As will be shown by our numerical results, the decreasing rank strategy is preferable for reasonably well-conditioned matrix completion problems, while the increasing rank strategy is more suitable for low-rank approximation problems where there does not exist a clear-cut desirable rank. Based on these observations, we use the decreasing rank strategy in the experiments of subsection 4.2, while the increasing rank strategy is used in

subsections 4.3–4.5.

4.2. Experiments on random matrix completion problems. The test matrices $M \in \mathbb{R}^{m \times n}$ with rank r in this subsection were created randomly by the following procedure (see also [22]): two random matrices $M_L \in \mathbb{R}^{m \times r}$ and $M_R \in \mathbb{R}^{n \times r}$ with i.i.d. standard Gaussian entries were first generated and then $M = M_L M_R^\top$ was assembled; then a subset Ω of p entries was sampled uniformly at random. The ratio $p/(mn)$ between the number of measurements and the number of entries in the matrix is denoted by "SR" (sampling ratio). The ratio $r(m+n-r)/p$ between the degree of freedom in a rank r matrix to the number of samples is denoted by "FR".

We first evaluate the sensitivity of LMaFit to the initial rank estimate K using the decreasing rank strategy of rank estimation. In this test, we used matrices with $m = n = 1000$ and $r = 10$. Three test cases were generated at the sampling ratios SR equal to 0.04, 0.08 and 0.3, respectively. In each case, we ran LMaFit for each of $K = 10, 11, 12, \cdots, 30$ on 50 random instances. The average number of iterations and average CPU time corresponding to this set of K values are depicted in Figures 4.1 (a) and (b), respectively. In these two figures, we observe a notable difference at the rank estimate $K = 10$ when the sampling ratio SR $= 0.04$. The reason is that at this low sampling ratio the rank estimate routine of LMaFit mistakenly reduced the working rank to be less than 10 and resulted in premature exists. For all other cases, we see that the number of iterations stayed at almost the same level and the CPU time only grew slightly as K increased from 10 to 30. Overall, we conclude that LMaFit is not particularly sensitive to the change of K value on this class of problems over a considerable range. Based on this observation, in all tests using the decreasing rank strategy, we set the initial rank estimate K either to $\lfloor 1.25r \rfloor$ or to $\lfloor 1.5r \rfloor$, where $\lfloor x \rfloor$ is the largest integer not exceeding x. Numerical results generated from these two K values should still be sufficiently representative.

FIG. 4.1. *The sensitivity* LMaFit *with respect to the initial rank estimation* K

(a) Iteration number

(b) CPU time in seconds
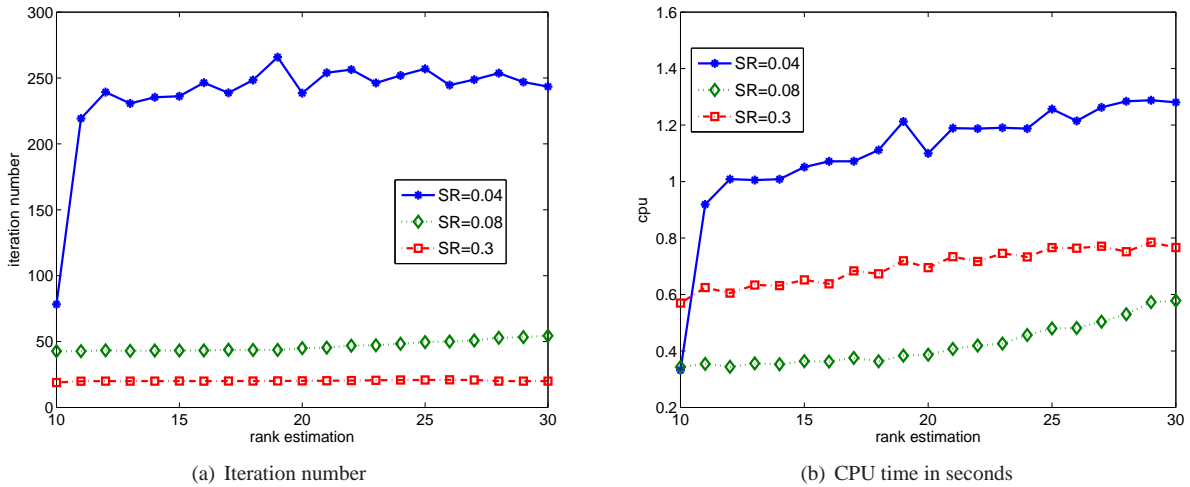
Our next test is to study the convergence bahavior of LMaFit with respect to the sampling ratio and true rank of M. In this test the dimension of M were set to $m = n = 1000$ and the initial rank estimate was set to $\lfloor 1.25r \rfloor$ in LMaFit. In Figure 4.2 (a), we plot the normalized residual $\|\mathcal{P}_\Omega(M - XY)\|_F / \|\mathcal{P}_\Omega(M)\|_F$ at all iterations for three test cases where the sampling ratio was fixed at SR $= 0.3$ and rank $r = 10, 50$ and 100, respectively. On the other hand, Figure 4.2 (b) is for three test cases where SR $= 0.04, 0.08$ and 0.3, respectively, while the rank was fixed at $r = 10$. Not surprisingly, these figures show that when the sampling ratio is fixed, the higher the rank is, the harder the problem is; and when the rank is fixed, the smaller the sampling ratio is, the harder the problem is. In all cases, the

convergence of the residual sequences appeared to be linear, but at quite different rates.

FIG. 4.2. *Convergence behavior of the residual in* LMaFit *runs*

(a) $m = n = 1000$, SR= 0.3

(b) $m = n = 1000$, $r = 10$
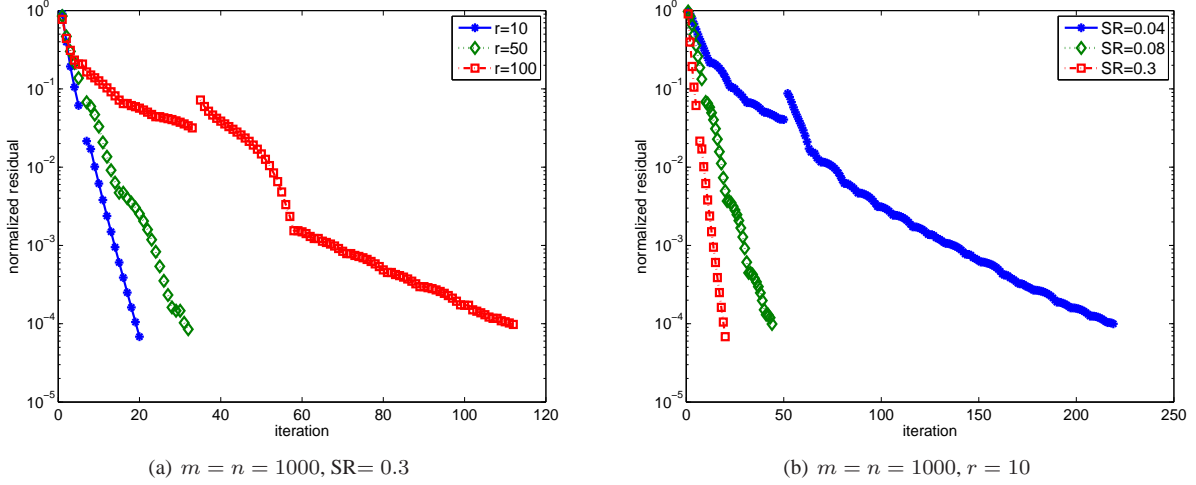
An important question about the factorization model (1.4) and our nonlinear SOR algorithm is whether or not our approach (model plus algorithm) has an ability in recovering low-rank matrices similar to that of solving the nuclear norm minimization model by a good solver. Or simply put, does our algorithm for (1.4) provide a comparable recoverability to that of a good nuclear norm minimization algorithm for (1.2) or (1.3)? We address this recoverability issue in the next test by generating phase diagrams in Figures 4.3 (a)-(b) for the two models (1.3) and (1.4), respectively. The solver FPCA [22] was chosen to solve (1.3) since it has been reported to have a better recoverability than a number of other nuclear norm minimization solvers. In this test, we used random matrices of size $m = n = 500$. We ran each solver on 50 randomly generated problems with the sampling ratio SR chosen in the order as it appear in $\{0.01, 0.06, 0.11, \cdots, 0.86\}$ and with each rank value $r \in \{5, 8, 11, \cdots, 59\}$. The two phase diagrams depict the success rates out of every 50 runs by each solver for each test case where a run was successful when the relative error $\|\mathcal{P}_\Omega(M - W)\|_F / \|\mathcal{P}_\Omega(M)\|_F$ between the true and the recovered matrices M and W was smaller than 10^{-3}. If a solver recovered all 50 random instances successfully for SR $= \alpha$ and $r = \beta$, then it ought to have equal or higher recoverability for SR $> \alpha$ and $r = \beta$. To expedite the part of the experiment involving FPCA, we chose to stop testing all other SR $> \alpha$ with $r = \beta$, and increment the r value. In Figures 4.3 (a)-(b), a white box indicates a 100% recovery rate, while a black box means a 0% rate. The parameter setting for LMaFit was $\texttt{tol} = 10^{-4}$, $K = \lfloor 1.25r \rfloor$ and $\texttt{est\_rank} = 1$, while for FPCA it was $\texttt{tol} = 10^{-4}$ and $\mu = 10^{-4}$. All other parameters were set to their respective default values. The two phase diagrams indicate that the recoverability of LMaFit is marginally inferior to that of FPCA in this experiment. Given the reported better recoverability of FPCA, it is reasonable to infer that the recoverability of LMaFit is comparable to those of the other nuclear norm minimization solvers studied in [22].

To have a quick assessment on the speed of LMaFit relative to those of other state-of-the-art solvers, we compared LMaFit with two nuclear norm minimization solvers, APGL and FPCA, and with a matlab version of OptSpace that solves a factorization model similar to ours but uses an SVD-based initial guess, on a set of small problems with $m = n = 1000$. The parameter setting for LMaFit was the same as in the previous experiment. In particular, the decreasing rank strategy was used. The parameter μ for the model (1.3) was set to $10^{-4}\sigma$ as suggested by the testing scripts in the package APGL, where σ is the largest singular value of $\mathcal{P}_\Omega(M)$. The stopping tolerance for all solvers was set to 10^{-4} and all other parameters were set to their default values. A summary of

FIG. 4.3. *Phase diagrams for matrix completion recoverability*

(a) Model (1.3) solved by `FPCA`

(b) Model (1.4) solved by `LMaFit`
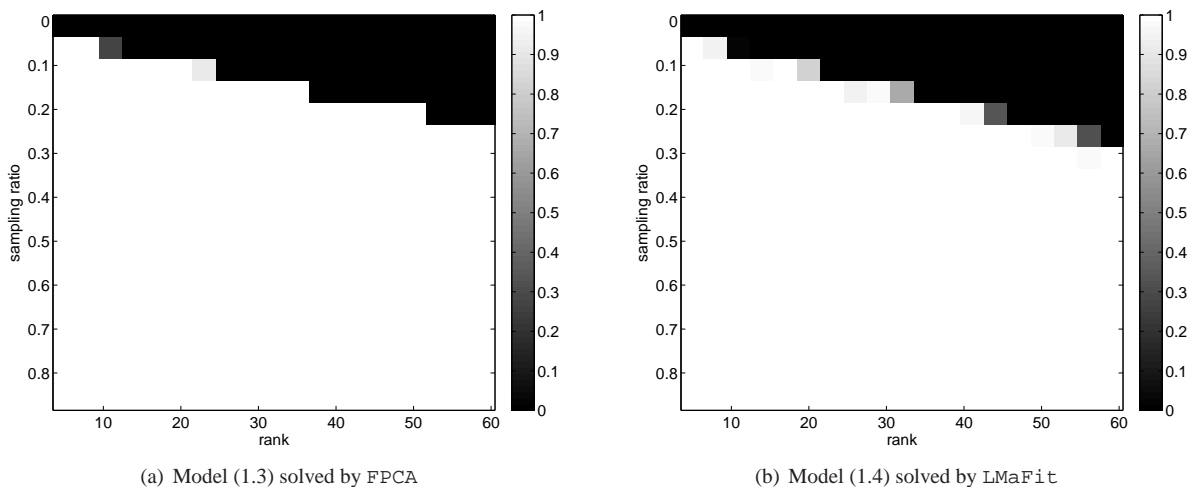
the computational results is presented in Table 4.1, where "time" denotes the CPU time measured in seconds and rel.err $:= \|W - M\|_F/\|M\|_F$ denotes the relative error between the true and the recovered matrices M and W ("tsvd" will be explained below).

TABLE 4.1
Comparison of four solvers on small problems with varying rank and sampling ratio

Problem			APGL				FPCA			OptSpace		LMaFit			
												$K = \lfloor 1.25r \rfloor$		$K = \lfloor 1.5r \rfloor$	
r	SR	FR	μ	time	rel.err	tsvd	time	rel.err	tsvd	time	rel.err	time	rel.err	time	rel.err
10	0.04	0.50	5.76e-03	3.89	4.04e-03	82%	32.62	8.21e-01	12%	109.75	8.06e-03	0.98	4.72e-04	1.00	4.35e-04
10	0.08	0.25	1.02e-02	2.25	6.80e-04	71%	13.24	7.30e-04	19%	91.52	7.13e-05	0.35	2.27e-04	0.40	2.19e-04
10	0.15	0.13	1.78e-02	2.44	2.14e-04	66%	7.76	4.21e-04	42%	40.06	1.94e-05	0.39	1.16e-04	0.41	1.48e-04
10	0.30	0.07	3.42e-02	4.11	1.40e-04	58%	17.54	1.97e-04	72%	35.99	3.90e-05	0.59	8.99e-05	0.62	9.91e-05
50	0.20	0.49	2.94e-02	123.90	2.98e-03	93%	71.43	4.64e-04	56%	4114.14	6.34e-04	3.96	3.03e-04	4.96	2.63e-04
50	0.25	0.39	3.59e-02	23.80	8.17e-04	87%	101.47	3.24e-04	67%	3449.38	3.24e-05	2.98	1.89e-04	3.20	2.11e-04
50	0.30	0.33	4.21e-02	18.64	6.21e-04	85%	146.24	2.64e-04	75%	3139.45	2.64e-05	2.56	1.78e-04	2.78	1.91e-04
50	0.40	0.24	5.53e-02	19.17	3.69e-04	82%	42.28	2.16e-04	77%	3316.98	2.30e-05	2.28	1.11e-04	2.69	1.65e-04
100	0.35	0.54	5.70e-02	73.48	1.24e-03	92%	259.37	5.41e-04	77%	13.14	9.92e-01	13.07	3.01e-04	17.40	3.09e-04
100	0.40	0.47	6.37e-02	63.08	8.19e-04	91%	302.82	4.11e-04	79%	13.25	9.91e-01	9.74	2.56e-04	11.39	2.41e-04
100	0.50	0.38	7.71e-02	61.44	4.91e-04	90%	359.66	3.10e-04	82%	112.24	9.21e-01	7.30	1.55e-04	7.37	1.92e-04
100	0.55	0.35	8.40e-02	50.78	4.12e-04	89%	360.28	2.89e-04	81%	111.66	9.19e-01	6.23	1.14e-04	7.18	9.99e-05

From Table 4.1, we see that `LMaFit` is at least several times (often a few orders of magnitude) faster than all other solvers to achieve a comparable accuracy. We note that the accuracy of the solver `OptSpace` on problems with rank 100 could not be improved by using a smaller tolerance. Of course, the reported performances of all the solvers involved were pertinent to their tested versions under the specific testing environment. Improved performances are possible for different parameter settings, on different test problems, or by different versions. However, given the magnitude of the timing gaps between `LMaFit` and others, the speed advantage of `LMaFit` should be more than evident on these test problems. (We also tested `LMaFit` with the increasing rank strategy and found that it was not as effective as the decreasing rank strategy on these random matrix completion problems.)

In Table 4.1, the item "tsvd" is the percentage of CPU time spent on SVD-related calculations, as estimated by the MATLAB profiler and obtained from separate runs. As can be seen, for `APGL` and `FPCA` SVD-related calculations essentially dominate their total costs (with the exception of extremely low-rank cases for `FPCA`). On the other hand, for

`LMaFit`, the total cost is dominated by low-rank or sparse matrix to matrix multiplications (which are also required by other solvers), while the cost of solving the least squares problem in (2.11b) is either negligible or at most 11% of the total CPU time. Therefore, avoiding SVD-related calculations is a main reason why `LMaFit` is much faster than the nuclear norm minimization solvers `APGL` and `FPCA`, validating our original motivation of solving the factorization model.

The next test was on large-scale random matrix completion problems in which we compared `LMaFit` with `APGL` following the experiment setup given in section 4.2 of [29]. The other solvers `FPCA` and `OptSpace` were excluded from this comparison since they would have demanded excessive CPU times. Summaries of the computational results are presented in Table 4.2 for noiseless data and Table 4.3 for noisy data, where both the noiseless and noisy data were generated as in [29]. In these two table, "iter" denotes the number of iterations used, and "#sv" denotes the rank of the recovered solution. The statistics contained in these two tables verify two key observations: (a) solving the factorization model is reliable for matrix completion on a wide range of problems, and (b) our nonlinear SOR algorithm, as implemented in `LMaFit`, has a clear speed advantage in solving many large-scale problems.

TABLE 4.2
Numerical results on large random matrix completion problems without noise.

Problem				APGL					LMaFit ($K = \lfloor 1.25r \rfloor$)				LMaFit ($K = \lfloor 1.5r \rfloor$)			
n	r	SR	FR	μ	iter	#sv	time	rel.err	iter	#sv	time	rel.err	iter	#sv	time	rel.err
1000	10	0.119	0.167	1.44e-2	39	10	2.47	3.04e-4	23	10	0.29	1.67e-4	23	10	0.28	1.73e-4
1000	50	0.390	0.250	5.36e-2	40	50	14.48	3.08e-4	18	50	1.88	6.58e-5	18	50	2.04	7.62e-5
1000	100	0.570	0.334	8.58e-2	53	100	49.67	3.98e-4	20	100	5.47	1.86e-4	21	100	5.99	1.42e-4
5000	10	0.024	0.166	1.37e-2	52	10	12.48	2.17e-4	29	10	1.99	1.71e-4	29	10	2.17	1.77e-4
5000	50	0.099	0.200	6.14e-2	76	50	161.82	1.26e-3	20	50	15.87	2.72e-5	20	50	16.49	3.86e-5
5000	100	0.158	0.250	1.02e-1	60	100	316.02	3.74e-4	26	100	57.85	1.57e-4	27	100	60.69	1.47e-4
10000	10	0.012	0.166	1.37e-2	53	10	23.45	3.61e-4	34	10	5.08	1.54e-4	34	10	5.56	1.66e-4
10000	50	0.050	0.200	5.97e-2	56	50	225.21	2.77e-4	23	50	44.80	4.76e-5	23	50	48.85	5.70e-5
10000	100	0.080	0.250	9.94e-2	71	100	941.38	2.87e-4	30	100	168.44	1.63e-4	30	100	176.45	1.70e-4
20000	10	0.006	0.167	1.35e-2	57	10	60.62	2.37e-4	38	10	12.46	1.44e-4	38	10	13.60	1.57e-4
30000	10	0.004	0.167	1.35e-2	59	10	95.50	1.96e-4	39	10	20.55	1.71e-4	39	10	23.48	1.73e-4
50000	10	0.002	0.167	1.35e-2	66	10	192.28	1.58e-4	42	10	43.43	1.81e-4	42	10	49.49	1.84e-4
100000	10	0.001	0.167	1.34e-2	92	10	676.11	2.10e-4	46	10	126.59	1.33e-4	46	10	140.32	1.30e-4

TABLE 4.3
*Numerical results on large random matrix completion problems with noise.*

Problem				APGL					LMaFit ($K = \lfloor 1.25r \rfloor$)				LMaFit ($K = \lfloor 1.5r \rfloor$)			
n	r	SR	FR	μ	iter	#sv	time	rel.err	iter	#sv	time	rel.err	iter	#sv	time	rel.err
1000	10	0.119	0.167	1.44e-2	39	10	2.94	4.53e-2	18	10	0.24	4.53e-2	18	10	0.23	4.53e-2
1000	50	0.390	0.250	5.36e-2	39	50	13.73	5.51e-2	17	50	1.76	5.51e-2	17	50	1.93	5.51e-2
1000	100	0.570	0.334	8.59e-2	50	100	43.11	6.40e-2	17	100	4.70	6.40e-2	18	100	5.31	6.40e-2
5000	10	0.024	0.166	1.38e-2	46	10	12.71	4.51e-2	26	10	1.78	4.51e-2	26	10	2.04	4.51e-2
5000	50	0.099	0.200	6.14e-2	67	50	135.89	4.97e-2	19	50	15.05	4.97e-2	19	50	15.88	4.97e-2
5000	100	0.158	0.250	1.02e-1	49	100	223.73	5.68e-2	18	100	39.81	5.68e-2	18	100	42.83	5.68e-2
10000	10	0.012	0.166	1.37e-2	50	10	25.75	4.52e-2	29	10	4.41	4.52e-2	29	10	4.86	4.52e-2
10000	50	0.050	0.200	5.97e-2	51	50	187.84	4.99e-2	23	50	45.04	4.99e-2	23	50	49.61	4.99e-2
10000	100	0.080	0.250	9.95e-2	58	100	681.45	5.73e-2	22	100	127.01	5.73e-2	22	100	134.87	5.73e-2
20000	10	0.006	0.167	1.35e-2	53	10	57.64	4.53e-2	33	10	11.21	4.53e-2	33	10	13.27	4.53e-2
30000	10	0.004	0.167	1.35e-2	55	10	89.22	4.52e-2	34	10	17.63	4.52e-2	34	10	20.89	4.52e-2
50000	10	0.002	0.167	1.35e-2	58	10	173.07	4.53e-2	37	10	40.24	4.53e-2	37	10	45.97	4.53e-2
100000	10	0.001	0.167	1.34e-2	70	10	517.36	4.53e-2	40	10	115.27	4.53e-2	40	10	123.81	4.53e-2

4.3. Experiments on random low-rank approximation problems. We now consider applying matrix completion algorithms to randomly generated low-rank matrix approximation problems. The goal is to find a low-rank approximation to a mathematically full-rank matrix M whose singular values gradually tend to zero, though none

is exactly zero. Since there does not exist a "best low rank matrix" in such approximations, any evaluation of solution quality must take into consideration of two competing criteria: rank and accuracy. The only clear-cut case of superiority is when one solution dominates another by both criteria, i.e., a lower rank approximation with a higher accuracy.

In this experiment, all random instances of $M \in \mathbb{R}^{m \times n}$ were created as follows: two matrices $M_L \in \mathbb{R}^{n \times n}$ and $M_R \in \mathbb{R}^{n \times n}$ with i.i.d. standard Gaussian entries are first generated randomly; then M_L and M_R are orthogonalized to obtain U and V, respectively; finally the matrix $M = U \Sigma V^\top$ is assembled. Here Σ is a diagonal matrix whose diagonal elements σ_i, for $i = 1, \cdots, n$, are either the power-law decaying, that is, $\sigma_i = i^{-3}$, or the exponentially decaying, that is, $\sigma_i = e^{-0.3i}$. Hence, all singular values are positive, and there are 99 and 46 entries whose magnitude are greater than 10^{-6} in these two types of Σ, respectively. These diagonals are illustrated in Figures 4.4 (a) and (b). The sampling procedures are the same as in those 4.2. In this test, the dimension and rank of M were set to $n = 500$ and $r = 10$, respectively.

FIG. 4.4. *illustration of decaying partterns of the singular values σ*

(a) power-low decaying: $\sigma_i = i^{-3}$

(b) exponentially decaying: $\sigma_i = e^{-0.3i}$
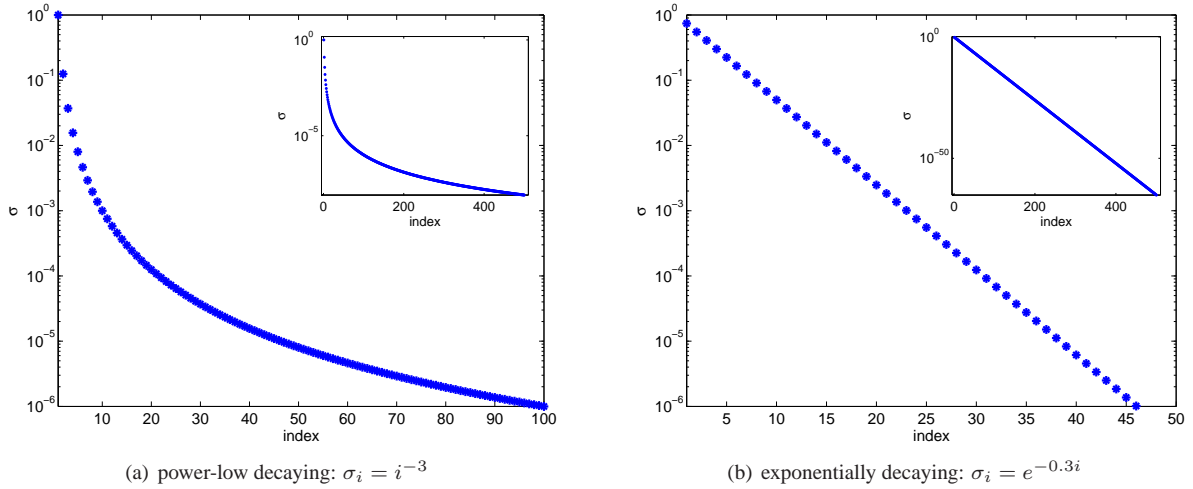
We compared `LMaFit` with the solvers `APGL` and `FPCA` [22]. The parameter μ for the model (1.3) was set to 10^{-4}. The stopping tolerance for all solvers was set to 10^{-4}. We set the parameters `truncation` $= 1$, and `truncation_gap` $= 100$ in `APGL`. For `LMaFit` with `est_rank` $= 1$, we set $K = 50$, and for `LMaFit` with `est_rank` $= 2$, we set $K = 1$, `rank_max` $= 50$ and `rk_inc` $= 1$. All other parameters were set to default values for the two solvers. A summary of the computational results is presented in Table 4.4. We can see that `LMaFit` with `est_rank` $= 2$ was significantly better than other solvers. The decreasing rank strategy of `LMaFit`, as it is currently implemented with `est_rank` $= 1$, is clearly not suitable for these low-rank approximation problems since there is no "true low rank" as in matrix completion problems. Specifically, this strategy (`est_rank` $= 1$) reduced the rank estimate too aggressively.

4.4. Experiments on "real data". In this subsection, we consider low-rank matrix approximation problems based on two "real data" sets: the Jester joke data set [7] and the MovieLens data set [12]. In these data set, only partial data are available from the underlying unknown matrices which are unlikely to be of exactly low rank. Nevertheless, matrix completion solvers have been applied to such problems to test their ability in producing low-rank approximations. As is mentioned above, an assessment of solution quality should take into consideration of both rank and accuracy. The Jester joke data set consists of four problems "jester-1", "jester-2", "jester-3" and "jester-all", where the

Problem		APGL				FPCA			LMaFit(est_rank=1)			LMaFit(est_rank=2)		
SR	FR	μ	#sv	time	rel.err	#sv	time	rel.err	#sv	time	rel.err	#sv	time	rel.err
							power-low decaying							
0.04	0.99	1.00e-04	90	16.30	6.48e-01	1	40.49	1.39e-01	5	0.70	3.68e-01	11	0.31	8.96e-03
0.08	0.49	1.00e-04	85	19.95	2.00e-01	2	45.81	4.38e-02	5	1.59	2.20e-01	20	0.60	1.13e-03
0.15	0.26	1.00e-04	7	1.73	4.05e-03	4	14.46	1.78e-02	5	1.47	1.52e-01	20	0.75	4.57e-04
0.30	0.13	1.00e-04	11	1.85	1.86e-03	4	31.48	1.04e-02	5	3.20	8.12e-02	22	1.33	2.36e-04
							exponentially decaying							
0.04	0.99	1.00e-04	100	15.03	7.50e-01	14	35.79	5.05e-01	5	0.48	3.92e-01	16	0.86	4.08e-01
0.08	0.49	1.00e-04	100	21.60	3.31e-01	8	39.82	1.24e-01	5	0.44	2.66e-01	26	1.84	1.98e-02
0.15	0.26	1.00e-04	100	17.43	4.71e-02	13	12.31	2.76e-02	5	0.63	2.39e-01	28	1.62	7.26e-04
0.30	0.13	1.00e-04	42	9.50	3.31e-03	14	29.13	1.71e-02	6	1.03	1.71e-01	30	2.01	2.38e-04

last one is obtained by combining all of the first three data sets, and the MovieLens data set has three problems "movie-100K", "movie-1M" and "movie-10M". They are available at www.ieor.berkeley.edu/~Egoldberg/jester-data and www.grouplens.org, respectively. For LMaFit, we set the parameters to tol $= 10^{-3}$, est_rank $= 2$, $K = 1$, and rk_inc $= 2$. For APGL, the parameter setting was tol $= 10^{-3}$, truncation $= 1$, and truncation_gap $= 20$. In addition, the model parameter μ for APGL was set to $\mu = 10^{-4}$ which produced better solutions than choosing $10^{-3}\sigma$ as suggested by the testing scripts in the package APGL, where σ is the largest singular value of the sampling matrix. Moreover, we set the maximum rank estimate to 80 for the jester problems and to 100 for the MovieLens problems for both LMaFit and APGL by specifying their parameters rank_max or maxrank, respectively. We note that since the jester problems have only 100 columns, it is not meaningful to fit a matrix of rank 100 to a jester data set. Since the entries of a underlying matrix M are available only on an index set Ω, to measure accuracy we computed the Normalized Mean Absolute Error (NMAE) as was used in [7, 22, 29], i.e.,

$$\text{NMAE} = \frac{1}{(r_{\max} - r_{\min})|\Omega|} \sum_{(i,j) \in \Omega} |W_{i,j} - M_{i,j}|,$$

where $r_{\min}$ and $r_{\max}$ are the lower and upper bounds for the ratings. Specifically, we have $r_{\min} = -10$ and $r_{\max} = 10$ for the jester joke data sets and $r_{\min} = 1$ and $r_{\max} = 5$ for the MovieLens data sets. We tried using a part of the available data as was done in [29] and found that APGL generally returned solutions with slightly higher NMAE-accuracy but also higher ranks than those returned by LMaFit, creating difficulties in interpreting solution quality (though the speed advantage of LMaFit was still clear). Therefore, we only report numerical results using all the available data in Table 4.5, where "#asv" denotes the approximate rank of a computed solution defined as the total number of singular values exceeding 10^{-8}.

Problem		APGL						LMaFit				
name	m/n	μ	iter	time	NMAE	rel.err	#asv	iter	time	NMAE	rel.err	#asv
jester-1	24983/ 100	1.00e-04	49	140.21	2.21e-02	1.73e-01	80	117	44.30	2.50e-02	1.86e-01	78
jester-2	23500/ 100	1.00e-04	49	133.74	2.27e-02	1.74e-01	80	118	43.11	2.56e-02	1.87e-01	78
jester-3	24938/ 100	1.00e-04	37	56.74	1.63e-06	9.57e-05	80	235	28.60	4.06e-05	9.31e-04	43
jester-all	73421/ 100	1.00e-04	48	284.15	1.90e-02	1.62e-01	80	114	96.47	2.03e-02	1.65e-01	80
moive-100K	943/ 1682	1.00e-04	100	82.43	6.89e-04	1.26e-03	100	507	24.59	9.95e-04	2.07e-03	94
moive-1M	6040/ 3706	1.00e-04	61	152.97	6.64e-02	9.59e-02	100	190	60.25	6.78e-02	9.85e-02	92
moive-10M	71567/ 10677	1.00e-04	57	1639.86	7.83e-02	1.32e-01	100	178	637.27	7.59e-02	1.29e-01	100

As can be seen from Table 4.5, LMaFit and APGL obtained low-rank approximation matrices of comparable quality on the all the problems, while LMaFit ran more than twice as fast, and returned matrices of slightly lower

approximate ranks (except for "jester-all" and "movie-10M"). It is particularly interesting to compare the two solvers on problem "jester-3" for which LMaFit reported a solution of rank 43 while APGL of rank 80. Even with a much lower rank, the LMaFit solution is almost as accurate as the APGL solution. Finally, we comment that without proper rank restrictions, the jester problems do not appear to be good test problems for low-rank matrix approximation since the matrices to be approximated have only 100 columns to begin with. In fact, LMaFit with est_rank = 1 and K=100 was able to find "solutions" of rank 100 after one iteration whose NMAE is of order 10^{-16}.

4.5. Image and video denoising or inpainting. In this subsection we apply LMaFit and APGL to grayscale image denoising (similar to what was done in [22]) and to color video denoising of impulsive noise for visualizing solution quality. The task here is to fill in the missing pixel values of an image or video at given pixel positions that have been determined to contain impulsive noise. This process is also called inpainting, especially when the missing pixel positions are not randomly distributed. In their original forms, these problems are not true matrix completion problems, but matrix completion solvers can be applied to obtain low-rank approximations.

In the first test, the 512×512 original grayscale image is shown in Figure 4.5(a), and we truncated the SVD of the image to get an image of rank 40 in Figure 4.5(b). Figures 4.5(c) and 4.5(f) were constructed from Figures 4.5(a) and (b) by sampling half of their pixels uniformly at random, respectively. Figure 4.5(i) was obtained by masking 6.34% of the pixels of Figure 4.54(b) in a non-random fashion. We set the parameters tol = 10^{-3}, est_rank = 2, $K = 20$ and max_rank = 50 for LMaFit, and tol = 10^{-3}, truncation = 1, truncation_gap = 20 and maxrank = 50 for APGL. The recovered images of Figures 4.5(c), (f) and (i) are depicted in Figures 4.5 (d) and (e), (g) and (h), and (j) and (k), respectively. A summary of the computational results is shown in Table 4.6. In the table, rel.err denotes the relative error between the original and recovered images. From these figures and the table, we can see that LMaFit can recover the images as well as APGL can, but significantly faster.
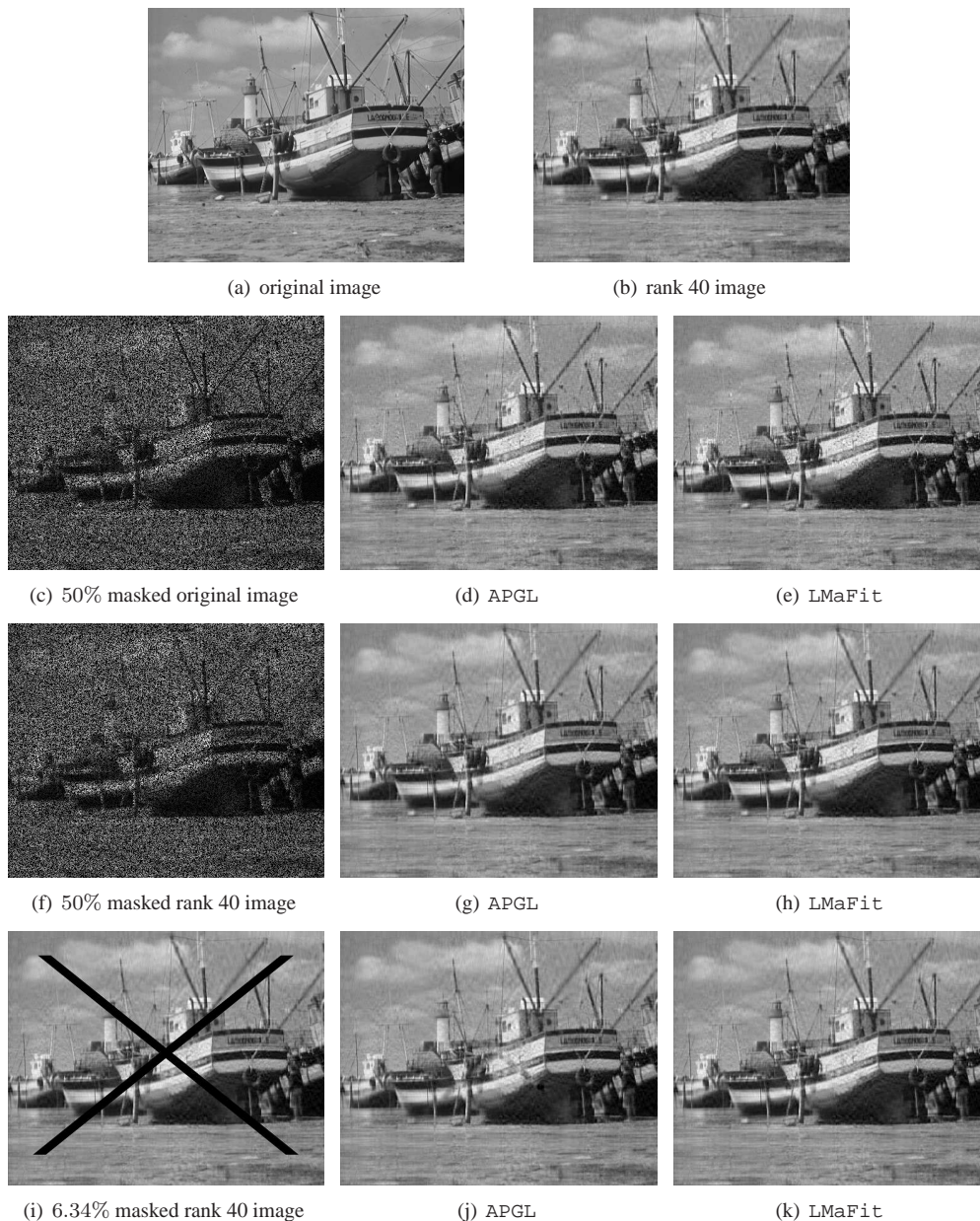
TABLE 4.6
Numerical results on image inpainting

problem		APGL					LMaFit			
image	r	μ	iter	#sv	time	rel.err	iter	#sv	time	rel.err
(c)	512	1.34e-02	38	50	5.28	8.92e-02	53	50	0.56	9.24e-02
(f)	40	1.34e-02	34	50	4.68	8.01e-02	42	40	0.43	7.94e-02
(i)	40	2.51e-02	32	50	5.02	9.07e-02	88	40	1.35	7.98e-02

Next, we apply LMaFit and APGL to fill in the missing pixels of a video sequence "xylophone.mpg" (available with the MATLAB Image Processing Toolbox). The video consists of p frames and each frame is an image stored in the RGB format, which is an m_r-by-n_r-by-3 cube. Here, $m_r = 240$, $n_r = 320$, and $p = 141$. The video was then reshaped into a $(m_r \times n_r)$-by-$(3 \times p)$, or 76800-by-423, matrix M. We sampled 50% pixels of the video uniformly at random. Three frames of the original video and the corresponding 50% masked images are shown in the first and second rows of Figure 4.6, respectively. We set the parameters tol = 10^{-3}, $K = 20$, rank_max = 80 and est_rank = 2 for LMaFit, and tol = 10^{-3}, truncation = 1, truncation_gap = 20 and maxrank = 80 for APGL. A summary of computational results is presented in Table 4.7 and the recovered images are shown in the third and fourth rows of Figure 4.6. From these figures, we can see that LMaFit was able to restore the static part of the video quite successfully, and the moving part of the video was still recognizable. Table 4.7 shows that APGL obtained a slightly higher accuracy than LMaFit did, but the latter was about 5 times faster in reaching the same order of accuracy.

We emphasize again that the purpose of the above image/video denoising or inpainting experiments was to visualize the solution quality for the tested algorithms, rather than demonstrating the suitability of these algorithms for the tasks of denoising or inpainting.

(a) original image

(b) rank 40 image

(c) 50% masked original image

(d) `APGL`

(e) `LMaFit`

(f) 50% masked rank 40 image

(g) `APGL`

(h) `LMaFit`

(i) 6.34% masked rank 40 image
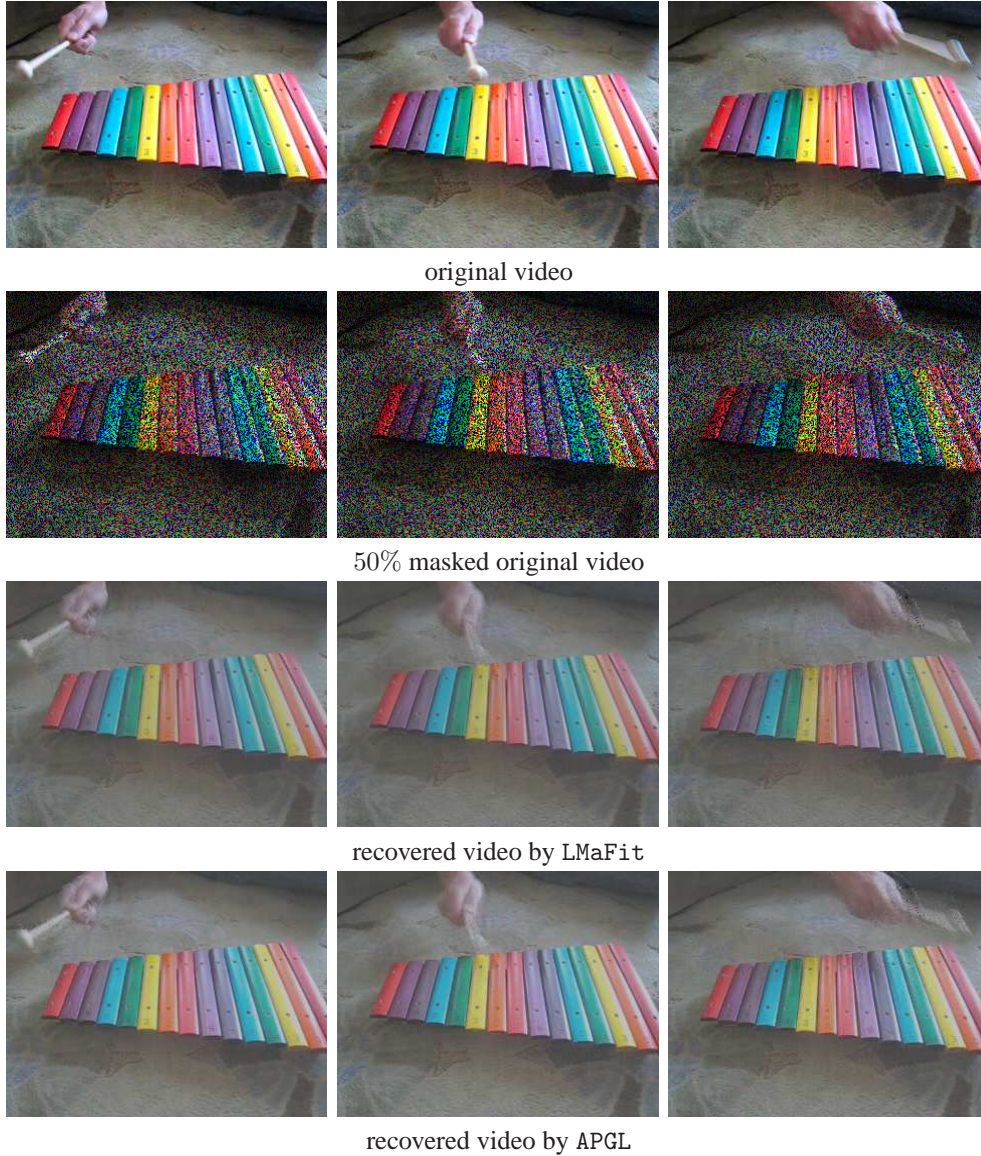
(j) `APGL`

(k) `LMaFit`

4.6. Summary of computational results. We performed extensive computational experiments on two classes of problems: matrix completion and low-rank approximation. On the completion problems, our nonlinear SOR algorithm, coupled with the decreasing rank strategy, has shown good recoverability, being able to solve almost all tested problems as reliably as other solvers. We do point out that randomly generated matrix completion problems are numerically well-conditioned with high probability. On the other hand, any solver, including ours, can break down in the face of severe ill-conditioning. On low-rank approximation problems where the concept of rank can be numerically blurry and the quality of solutions less clear-cut, our nonlinear SOR algorithm, coupled with the increasing rank strategy, has demonstrated a capacity of producing solutions of competitive quality on a diverse range of test problems.

Our numerical results, especially those on matrix completion, have confirmed the motivating premise for our

TABLE 4.7
Numerical results on video inpainting

problem		APGL					LMaFit			
video	m/n	μ	iter	#sv	time	rel.err	iter	#sv	time	rel.err
xylophone	76800/423	3.44e+01	34	80	516.22	4.58e-02	64	80	92.47	4.93e-02

FIG. 4.6. *Video Denoising*

original video

50% masked original video

recovered video by `LMaFit`

recovered video by `APGL`

approach that avoiding SVD-related calculations can lead to a much accelerated solution speed for solving matrix completion and approximation problems. Indeed, in our tests `LMaFit` has consistently shown a running speed that is several times, ofter a couple of magnitudes, faster than that of other state-of-the-art solvers.

5. Conclusion. The matrix completion problems is to recover a low-rank matrix from a subset of its entries. It has recently been proven that, by solving a nuclear-norm minimization model, an incoherent low-rank matrix can be exactly recovered with high probability from a uniformly sampled subset of its entries as long as the sample size is sufficiently large relative to the matrix sizes and rank. In this paper, we study the approach of solving a low-rank

factorization model for matrix completion. Despite the lack of a theoretical guarantee for global optimality due to model non-convexity, we have shown empirically that the approach is capable of solving a wide range of randomly generated matrix completion problems as reliably as solving the convex nuclear-norm minimization model. It remains a theoretical challenge to prove, or disprove, that under suitable conditions the low-rank factorization model can indeed solve matrix completion problems with high probability.

The main contribution of the paper is the development and analysis of an efficient nonlinear Successive Over-Relaxation (SOR) scheme that only requires solving a linear least-squares problem per iteration instead of a singular-value decomposition. The algorithm can be started from a rough over-estimate of the true matrix rank for completion problems, or started from a small initial rank (say, rank-1) for low-rank approximation problems. Extensive numerical results show that the algorithm can provide multi-fold accelerations over nuclear-norm minimization algorithms on a wide range of matrix completion or low-rank approximation problems, thus significantly extending our ability in solving large-scale problems in this area.

In order to solve large-scale and difficult problems, further research on rank estimation techniques is still needed to improve the robustness and efficiency of not only our algorithm, but also nuclear norm minimization algorithms that use partial singular value decompositions rather than full ones. Given the richness of matrix completion and approximation problems, different algorithms should be able to find usefulness in various areas of applications.

REFERENCES

[1] A. BECK AND M. TEBOULLE, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM Journal on Imaging Sciences, 2 (2009), pp. 183–202.

[2] E. J. CANDES AND Y. PLAN, *Matrix completion with noise*, tech. rep., Stanford University, 2009.

[3] E. J. CANDES AND B. RECHT, *Exact matrix completion via convex optimization*, Foundations of Computational Mathematics, (2009).

[4] E. J. CANDÈS AND T. TAO, *The power of convex relaxation: Near-optimal matrix completion*, tech. rep., Applied and Computational Mathematics, Caltech, 2009.

[5] W. DAI AND O. MILENKOVIC, *Set: an algorithm for consistent matrix completion*, tech. rep., Department of Electrical and Computer Engineering, University of Illinois, 2009.

[6] L. ELDÉN, *Matrix Methods in Data Mining and Pattern Recognition (Fundamentals of Algorithms)*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2007.

[7] K. GOLDBERG, T. ROEDER, D. GUPTA, AND C. PERKINS, *Eigentaste: A constant time collaborative filtering algorithm*, Inf. Retr., 4 (2001), pp. 133–151.

[8] D. GOLDFARB, S. MA, AND Z. WEN, *Solving low rank matrix completion problems efficiently*, tech. rep. Invited paper at the 47th Allerton Conference on Communication, Control and Computing, Illinois, 2009.

[9] G. H. GOLUB AND C. F. VAN LOAN, *Matrix computations*, Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, third ed., 1996.

[10] L. GRIPPO AND M. SCIANDRONE, *On the convergence of the block nonlinear Gauss-Seidel method under convex constraints*, Oper. Res. Lett., 26 (2000), pp. 127–136.

[11] D. GROSS, *Recovering low-rank matrices from few coefficients in any basis*, tech. rep., Leibniz University, 2009.

[12] J. L. HERLOCKER, J. A. KONSTAN, A. BORCHERS, AND J. RIEDL, *An algorithmic framework for performing collaborative filtering*, in SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, 1999, ACM, pp. 230–237.

[13] C. JIAN-FENG, E. J. CANDES, AND S. ZUOWEI, *A singular value thresholding algorithm for matrix completion export*, tech. rep., 2008.

[14] R. H. KESHAVAN, A. MONTANARI, AND S. OH, *Matrix completion from a few entries*, tech. rep., Stanford Unviersity, 2009.

[15] ———, *Matrix completion from noisy entries*, tech. rep., Stanford University, 2009.

[16] R. H. KESHAVAN AND S. OH, *A gradient descent algorithm on the grassman manifold for matrix completion*, tech. rep., Dept. of Electrical Engineering, Stanford University, 2009.

[17] R. M. LARSEN, *Propack: Software for large and sparse svd calculations*. `http://soi.stanford.edu/~rmunk/PROPACK`.

[18] K. LEE AND Y. BRESLER, *Admira: Atomic decomposition for minimum rank approximation*, tech. rep., University of Illinois, Urbana-Champaign, 2009.

[19] Y. LIU, D. SUN, AND K. C. TOH, *An implementable proximal point algorithmic framework for nuclear norm minimization*, tech. rep., National University of Singapore, 2009.

[20] Z. LIU AND L. VANDENBERGHE, *Interior-point method for nuclear norm approximation with application to system identification*, SIAM Journal on Matrix Analysis and Applications, 31 (2009), pp. 1235–1256.

[21] Z. Q. LUO AND P. TSENG, *On the convergence of the coordinate descent method for convex differentiable minimization*, J. Optim. Theory Appl., 72 (1992), pp. 7–35.

[22] S. MA, D. GOLDFARB, AND L. CHEN, *Fixed point and Bregman iterative methods for matrix rank minimization*, tech. rep., Department of IEOR, Columbia University, 2008.

[23] R. MAZUMDER, T. HASTIE, AND R. TIBSHIRANI, *Regularization methods for learning incomplete matrices*, tech. rep., Stanford University, 2009.

[24] R. MEKA, P. JAIN, AND I. S. DHILLON, *Guaranteed rank minimization via singular value projection*, tech. rep., University of Texas at Austin, 2009.

[25] T. MORITA AND T. KANADE, *A sequential factorization method for recovering shape and motion from image streams*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 19 (1997), pp. 858–867.

[26] B. RECHT, *A simpler approach to matrix completion*, tech. rep., University of Wisconsin, 2009.

[27] B. RECHT, M. FAZEL, AND P. A. PARRILO, *Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization*, tech. rep., 2007.

[28] G. W. STEWART, *On the continuity of the generalized inverse*, SIAM J. Appl. Math., 17 (1969), pp. 33–45.

[29] K.-C. TOH AND S. YUN, *An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems*, tech. rep., Department of Mathematics, National University of Singapore, 2009.

[30] C. TOMASI AND T. KANADE, *Shape and motion from image streams under orthography: a factorization method*, Int. J. Comput. Vision, 9 (1992), pp. 137–154.

[31] P. TSENG, *Dual ascent methods for problems with strictly convex costs and linear constraints: a unified approach*, SIAM J. Control Optim., 28 (1990), pp. 214–242.

[32] P. TSENG, *Convergence of a block coordinate descent method for nondifferentiable minimization*, J. Optim. Theory Appl., 109 (2001), pp. 475–494.

[33] J. YANG AND X. YUAN, *An inexact alternating direction method for trace norm regularized least squares problem*, tech. rep., Dept. of Mathematics, Nanjing University, 2010.

[34] X. YUAN AND J. YANG, *Sparse and low-rank matrix decomposition via alternating direction methods*, tech. rep., Dept. of Mathematics, Hong Kong Baptist University, 2009.

[35] Y. ZHANG, `LMaFit`: *Low-rank matrix fitting*, 2009. http://www.caam.rice.edu/ optimization/L1/LMaFit/.

[36] ——, *An alternating direction algorithm for nonnegative matrix factorization*, tech. rep., Rice University, 2010.

[37] Z. ZHU, A. M.-C. SO, AND Y. YE, *Fast and near–optimal matrix completion via randomized basis pursuit*, tech. rep., Stanford University, 2009.